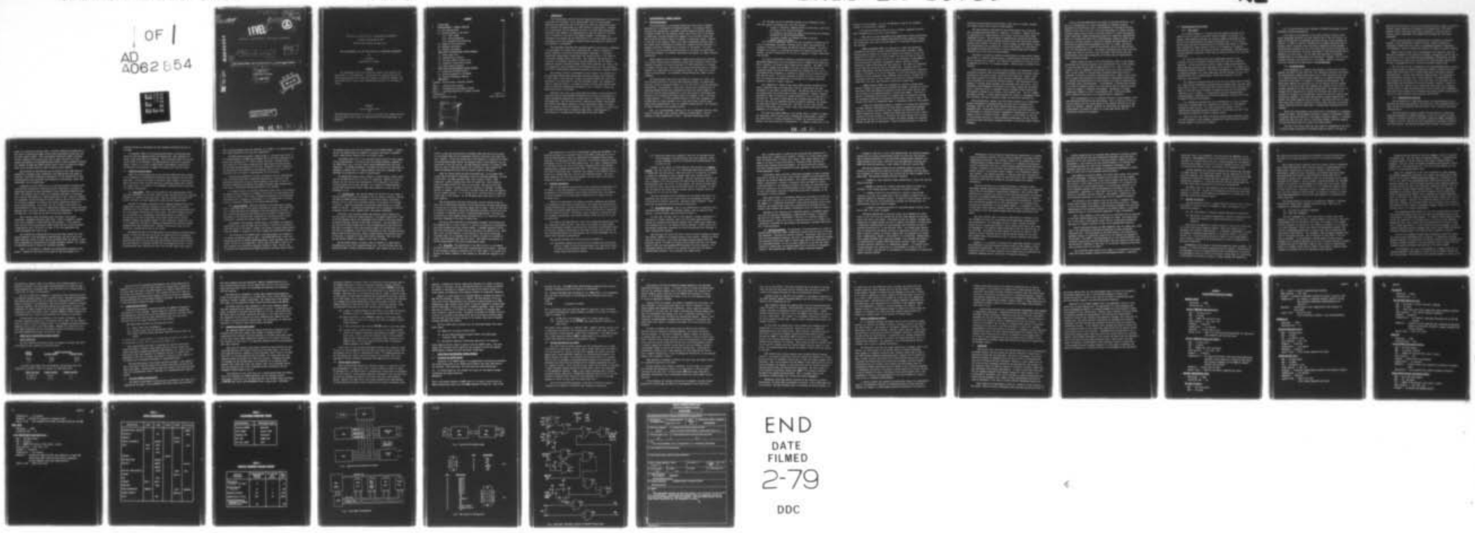
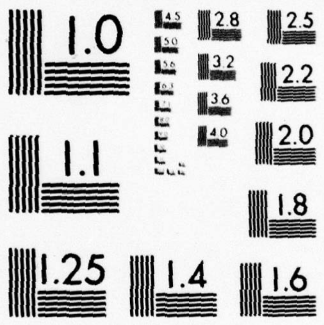
AD-A062 854 ROYAL AIRCRAFT ESTABLISHMENT FARNBOROUGH (ENGLAND) F/G 9/2
THE MICROPROCESSOR, AND ITS APPLICATION IN A LABORATORY ENVIRON--ETC(U)
AUG 77 M SOCKETT
UNCLASSIFIED RAE-TM-IT-165 DRIC-BR-59769 NL

1 OF 1
AD
A062854



END
DATE
FILMED
2-79
DDC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU

(19) BR59769

(14) RAE-TM-IT-165 TECH. MEMO

(18) DRIC

TECH. MEMO
IT 165

(1)

2

ROYAL AIRCRAFT ESTABLISHMENT

(9) Technical memo.,

(12) 47p.

(6)

THE MICROPROCESSOR, AND ITS APPLICATION IN A LABORATORY ENVIRONMENT.

by

(10) M. Sockett
(University of Surrey)

(11) August 1977

ROYAL AIRCRAFT ESTABLISHMENT

Technical Memorandum IT 165

Received for printing 26 August 1977

THE MICROPROCESSOR, AND ITS APPLICATION IN A LABORATORY ENVIRONMENT

by

M. Sockett

(University of Surrey)

SUMMARY

This Memorandum records the experience gained in the selection and application of a microprocessor. Observations are included about microprocessors generally, with particular emphasis on eight-bit machines. The Intel 8080A device, and the Intel SDK-80 Evaluation Kit, are considered in detail.

(This Memorandum was written as a result of a period of six months Industrial training, spent by the author at Instrumentation & Trials Department, RAE Bramshot.)

CONTENTS

1 INTRODUCTION

The microprocessor has been the most rapidly developing area of electronics during the last few years, and it seems probable that its revolutionary appearance and growth will now be followed by a period of consolidation which will establish its significance in a wide variety of applications. Although the microprocessor has the undoubted potential to replace many conventional hard-wired logic systems, the implementation of any task requires a different area of expertise on the part of the system designer. Accordingly, the advantages to be gained, and the difficulties to be overcome, when using microprocessors need to be thoroughly evaluated before any large-scale adoption of such systems is contemplated.

This Memorandum describes the experience gained in the process of applying a microprocessor to a simple data processing task. Section 2 comprises a general synopsis of the field of microprocessor technology. It describes generalised microprocessor architecture, and introduces some of the terms and definitions commonly used in this field. The categories into which microprocessors are normally classified are briefly explained, and the ranges of commercial products currently marketed within these categories are listed. The most widespread class of machines, 8-bit microprocessors, are then considered, and several representative contemporary systems are described in greater detail.

The problems involved in development of both hardware and software for a microprocessor system, particularly when starting initially, are considered at some length. In particular, a range of evaluation systems for 8-bit microprocessors - the first step for most microprocessor users - are analysed in terms of practical details. An attempt is made to provide some general guidelines to assist the inexperienced purchaser of such a machine to choose the system best suited to his needs.

Sections 3 and 4 are a more specialised description of work performed in connection with the Intel SDK-80 microprocessor evaluation kit purchased for study, principally concerned with its application to a typical data handling/processing task. The two sections cover, respectively, hardware development, and software development, during the initial experimental stages of working with this typical, small 8-bit machine. This part of the report is likely to be of detailed interest only to readers already using, or anticipating the acquisition of, a microprocessor system based on the Intel 8080A.

2 MICROPROCESSORS - GENERAL OVERVIEW

2.1 The Microprocessor

The Microprocessor is an advanced large scale logic device, designed principally as a replacement for hardwired systems. It resembles, both in structure and application, a low-speed, limited capability digital computer. A computer differs from the familiar, simple logic device only in its degree of complexity; each may ultimately be regarded as an electronic "black box" which produces particular outputs in response to appropriate inputs. However, in the case of the computer, there is no fixed correspondence between input and output; their relationship is defined by a variable internal structure, or program, previously configured by the operator of the system. It is this capacity to vary the operating characteristics of the device by modification of the stored program, or software, which gives the computer its almost infinite flexibility.

Rapid progress in the semiconductor manufacturing industry has resulted in a current state-of-the-art described as Large Scale Integration (LSI), which commonly corresponds to circuitry equivalent to 15000 transistors on a single 6 mm square silicon chip. This allows the logical processing power of a simple computer to be implemented on one integrated circuit. The result is a device termed a microprocessor - or preferably, microprocessing unit (MPU) to avoid confusion.

The microprocessor may thus be described as a computer on a chip - but this is rarely justified. Like any other computing system, a microprocessor must have access to peripheral devices if it is to be able to use its processing capabilities. In particular, it must have a memory store containing both its operating program and any data, intermediate results and system parameters which are required during running; and it must have means of interfacing with any parts of its working system which require an exchange of data - including a human operator. It is not currently possible to build such equipment into the MPU itself in useful amounts, or in a way which allows flexible usage. Single chip microprocessors - that is, complete working systems - are available, but their capabilities are severely limited.

Normally, a microprocessor consists of a set of integrated circuits, built around a central MPU, which together comprise a working system. Such an assembly is shown schematically in Fig 1. The basic constituents are:-

(a) The MPU, which will generally consist of one integrated circuit. Like any computer processing unit, this will contain,

- a group of registers, temporary stores in which data, results and locations are held during processing,
- the arithmetic and logic unit (ALU), which performs the necessary arithmetic and Boolean manipulations,
- circuitry to handle the transfer of data in and out,
- overall control circuitry.

(b) Parallel input/output ports; this role is normally satisfied by the use of a programmable peripheral interface device (PPI), which handles the exchange of parallel data between the MPU and external equipment. The PPI is designed to be capable of accepting any normal data handling convention, and assumes a suitable configuration at run time in response to simple instructions from the MPU.

(c) Serial input/output capability is generally provided by a universal synchronous/asynchronous receiver-transmitter device (USART) which enables the MPU to interface with normal serial communications channels. The USART is fed with a simple program from the MPU at run time, to define the formats and conventions under which serial data obtained from external devices, such as telecommunications equipment, is converted to parallel data as required by the MPU, and vice versa. It is also necessary to define the rate at which data must be processed by the USART; this is performed by an external Baud Rate Generator, composed of simple circuitry.

(d) Read/write or random access memory (RAM) provides the MPU with an easily accessible means of storing data, results, locations and any other information it may require during operation. The RAM may also be used to hold the program which the microprocessor follows in operation. However, standard semiconductor RAM is volatile; all stored information is lost if the device is switched off. The amount of RAM required by a microprocessor system may be built up from a set of integrated circuits of specific capacity. For example, the most popular RAM IC in current use - the Intel 2111-A4 - has a memory capacity of 256 words, each of 4 bits. Thus, eight such devices may be used to implement a 1K x 8 memory bank.

(e) Read only memory (ROM) is, like the MPU itself, a product of modern LSI techniques, and is equally vital in making the microprocessor a feasible and useful device. Storing information in a ROM is an involved procedure; however, once stored the data is non-volatile - it remains in place even in the absence of a power supply. As such, the ROM may be used for the permanent storage of microprocessor programs.

A microprocessor ROM may consist of one or several integrated circuits, which in general will be of one of three types:-

- ROM, which are mask-programmed by the manufacturer during construction; this involves all the initial expense associated with mass production methods, but is obviously ideal when very large numbers of a particular stored pattern are required,

- PROM (programmable ROM), are supplied in a blank form and may be programmed by the user as needed, using relatively simple equipment. PROM devices normally consist of an array of nichrome fuse elements, in which the appropriate bit pattern is set up by burning out correctly selected fuses to differentiate between logical "0"s and "1"s. The advantage of using PROMs is the ease with which they may be programmed; however, they have several undesirable features which may make them unpopular. The method of programming is such that the process may only be performed once - programs cannot be changed subsequently, and any mistakes made during programming are irreparable. There are also doubts as to the faithfulness of stored data (quoted figures of 98-99% are clearly inadequate when a PROM may contain several thousand bits of information) and there is some evidence that burned out nichrome fuses may "grow back" with time, thus randomly altering the recorded data.

- EPROM (erasable PROM), are programmed by electronic methods which require slightly more sophisticated equipment than PROMs. Although some doubts have been raised concerning the long-term reliability of these devices, they are not prone to the inherent problems of fusible-link PROMs. Correction of errors, or subsequent modifications of programs, may be performed by erasing the existing bit pattern by irradiation with suitable ultraviolet light. The silicon chip itself is encapsulated under a transparent window for this purpose.

A complete microprocessor system is composed of an appropriate combination of these elements, interconnected by three parallel communications pathways. These are: firstly, the data bus, which transfers the operational data and results between the devices, secondly the address bus, which is used to select required locations in memory, or particular data paths through the I/O device, and thirdly the control bus, which channels the system commands and responses between the MPU and its peripherals. To get the system working, all that are required are power supplies, and normally a clock input to the MPU, provided by a simple circuit built around a timing crystal.

The vast potential represented by the microprocessor lies in the fact that the system described above is a complete - if limited - computer which may be constructed on a single printed circuit board, from no more than a dozen integrated circuits. The semiconductor manufacturers, through the popularity of the electronic calculator, digital wristwatch and other solid-state consumer products, have learned to produce LSI circuits cheaply in large numbers. Hence, the microprocessor can be produced at a price which ensures it a mass market; any automatic machine which requires a control mechanism may now be computer controlled by its own built-in microprocessor.

The result is potential hardware standardisation; the same control system may be used in a domestic washing machine, a sophisticated laboratory instrument, and an automated steel foundry. The individual requirements of each job can be accommodated by the use of a suitable program - variation may be limited to the software.

The established trend in control system design is the replacement of the traditional mechanical and electromechanical equipment with solid-state digital electronics of far greater speed, reliability and capability. However, hard-wired logic systems of this type require enormous development effort, which must be repeated for each new and different product. The requirements must be analysed, a suitable system designed, and prototypes built and tested. Almost inevitably, some debugging is necessary at this stage; finding and rectifying faults - which may be due to component failure, mistakes during construction, inaccuracies in design or even errors in the original logic - can be a lengthy and arduous process. As these systems become more complex, so debugging becomes more difficult, and there is greater leeway for operational unreliability.

The microprocessor, however, can allow these problems to be circumvented. Hardware development is straightforward, and need only be performed once, providing a stock of "off the shelf" systems which may be employed for all subsequent jobs. The problems involved in a particular task are centred in the development of suitable software; programming a microprocessor, as with any computer, can be a difficult process. However, it is not subject to the sort of uncertainties that seem to plague hardware implementation, and the computer industry has well-established techniques available to facilitate simple software debugging.

This is the philosophy behind the concept of the microcontroller - the use of a microprocessor as a direct replacement for systems currently implemented with hard-wired logic, electromechanical or mechanical devices. Central to this idea is the use of the read-only-memory for program storage; although computer control may be a desirable feature in a great many applications, traditional computer programming methods are completely inappropriate. There is no large-scale market for a microprocessor controlled machine which requires an on-line terminal, a high-level language operating system, and which must be appropriately programmed every time it is used. The microcontroller program can be written by specialists, then set in ROM, and thus built into the production unit. Far greater flexibility can be built into the program than could be provided in any traditional logic system; and major modifications, should they subsequently be desired, may be implemented by simply replacing the ROM.

However, beyond the area of the microcontroller, where it may advantageously replace existing mechanisms, the microprocessor has a broad spectrum of potential applications, many of them peculiar to it. Used as a microcomputer, that is, programmed individually by the user to suit a particular task, it may be used for the many small processing tasks which would be uneconomic to perform on existing computing facilities. There is also a great deal of simple work currently run on main-frame or minicomputers which does not justify the sophistication of these machines, and might be completed more efficiently using a microprocessor. A microcomputer is sufficiently flexible to be used as a general purpose tool, which may be used for many varied jobs within, for example, a laboratory environment. Conversely, it becomes economically feasible to build a microcomputer into a particular system, where constant availability of its capacity might be convenient.

The microprocessor is not a miraculous solution to every problem; it has many associated difficulties and shortcomings. Some of these are due to its relative newness, and the unfamiliarity of the concepts involved in its use; others are inherent. However, provided its limitations are understood, the microprocessor is potentially useful in a great many fields, and its capabilities should not be ignored.

2.2 Microprocessors on the market

2.2.1 Word length

It is common practice to classify computing machines according to the "width" in bits of parallel data which they are designed to process. For example, large mainframe computers are commonly capable of handling 32 bits, and the more recently popular minicomputers generally have a 16 bit format. Clearly, high precision calculations may be performed much faster on a machine with a longer word length; however, this is at the expense of far greater complexity and cost, and thus requires definite justification. Microprocessors are also primarily classified in this way, and available devices fall into four categories:-

(a) Four-bit machines, which are in many ways identical to the electronic calculator integrated circuits from which they were originally developed. As such they are well suited to simple control applications and number handling operations where high speed is not required - for example, in a man/machine interface situation. The limited performance of a four-bit machine is completely adequate for a great many uses, and the extremely low unit cost - typically £5-£10 for a minimum system - makes them attractive for mass production incorporation.

(b) Eight-bit microprocessors are by far the most popular type in current use. They provide considerable flexibility and processing power, and their byte-sized format (1 byte = 8 bits) makes them ideally fitted to data handling roles. Eight bit machines can be configured to perform any task from an intelligent controller, up to a complete, if comparatively slow and unsophisticated, computing system. Depending on the required complexity, a system built around an eight-bit MPU may cost £50-£500.

(c) Sixteen bit machines are true microcomputers; at a cost of around £3000 such a microprocessor will commonly be as powerful as a minicomputer costing ten times as much. It is thus economically feasible to have a sophisticated computing capability dedicated to a particular job, and to have far greater availability of general computer access.

There are also a few twelve-bit format microprocessors available, which are best included in this category, although their capabilities are normally limited compared to sixteen-bit machines.

(d) Bit slice devices, which represent a different philosophy in microprocessors, requiring explanation.

A bit slice is an integrated circuit which contains a section of the actual processing circuitry of an MPU, normally two or four bits "wide". An eight bit ALU may thus be configured from four, two-bit slices, connected in parallel. In order to produce a working MPU, the primary additional need is suitable control circuitry; since this may be designed to provide whatever functions may be desired, the use of bit slices thus allows the production of an MPU exactly tailored to the user's requirement. Such specialisation may be essential in certain cases - such as where very high speed is necessary - although there is the disadvantage that variations in hardware, with all their attendant problems, again become part of the design. However, this may be partially overcome by the application of microprogramming.

2.2.2 Microprogramming

Microprogramming is a technique most commonly utilised with bit-slice systems, but which has been used with a few externally controlled 16-bit MPUs. The normal general purpose microprocessor has a fixed instruction set; that is, a definite number (typically about eighty for an eight-bit machine) of instruction codes which it is capable of decoding and implementing. The codes which comprise the instruction set are the smallest units of program to which the user has access, and a program will normally be built up of a suitable combination of them, performed sequentially. The instruction set is designed by the manufacturer to provide the functions most useful to the operator; for example, all microprocessor instruction sets will contain codes allowing the movement of data among the registers, between registers and memory, and between registers and I/O devices. Similarly there will be codes which facilitate arithmetic operations, Boolean logic functions, and internal control operations such as conditional or unconditional branching within the program.

Within the microprocessor, each instruction is implemented as a miniature program in itself. The processing circuitry is capable of performing a handful of specific operations familiar to any digital electronic system. A particular instruction code will give rise to a previously defined, hard-wired sequence of these operations which will produce the desired effect.

The use of bit slices allows the user access to programming at the level of these internal operations; by suitably wiring his control circuitry, he may define the instruction set which the microprocessor will employ, and include whatever unusual functions are necessary to the application. This control hardware, rather than being constructed individually to suit the task in question, may be implemented from standard devices which have been microprogrammed.

Since each code within the instruction set is treated as a miniature program, it may conveniently be handled as such. Control circuitry, on receiving an instruction code, finds a corresponding sequence of operations - a microprogram - stored in a ROM, and follows it to produce the desired results.

There is thus a double level of development to be completed; first the MPU must be assembled in the appropriate hardware configuration, and the set of instructions it will use defined by microprogramming. Subsequently, this MPU may be treated as a single unit, and built into a microprocessor system which in turn is programmed to act as a device with definite capabilities.

The use of bit slices in combination with microprogramming makes available to the logic designer an approach intermediate between a hardware-orientated system (hard wired logic) and a software-based system (the general purpose microprocessor). As such, this technique combines many of the advantages of both alternatives, and also some of their drawbacks. Microprogrammed bit slice systems are currently in use in two main areas; for high speed control applications, and for emulation of existing mainframe computers. However, it is very likely that they will find a great many varied uses in the immediate future; because of their specialised nature, which requires expert management, these devices have tended to be overshadowed by general purpose microprocessors. Only now is their potential being recognised and explored.

2.2.3 Manufacturing technology

The characteristics of any microprocessor are largely governed by the manufacturing technology employed in the production of the integrated circuits. There are two broad categories, MOS and bipolar, each of which has particular characteristics.

The vast majority of general purpose microprocessors are constructed using metal-oxide/silicon (MOS) technology, in which the circuitry consists primarily of arrays of unipolar field-effect transistors (FETs) configured on a "chip" of silicon. This was the first semiconductor technology to produce true LSI circuits. The earliest successful MOS circuitry - the simplest to master on a practical production basis - utilised conduction processes due to the motion of positively-charged "holes" within semiconductor materials, and was hence termed "p-MOS". Later developments introduced "n-MOS" circuitry, in which conduction by electrons gave rise to faster operation and lower energy requirements; and complementary MOS (C-MOS) in which integrated circuits are composed of p-MOS and n-MOS elements "back to back", allowing even higher speed, more efficient operation, and far greater noise immunity.

MOS circuitry is characterised by low production cost, high levels of integration, economical power consumption, and comparatively slow speed of operation. Thus, MOS technology is appropriate for the construction of most microprocessors, where operation times measured in microseconds, rather than nanoseconds, are no disadvantage.

Integrated circuits before the era of LSI were normally configured with circuitry built around more traditional bipolar transistors, utilising the technology pioneered by Texas Instruments, termed transistor/transistor logic (TTL). The primary virtue of TTL is its extremely high speed of operation, but it has not been found to be ideally applicable to LSI. Levels of integration comparable to MOS circuitry are hard to achieve, and the resulting devices are relatively expensive to produce, and require considerable supplies of power during operation. As a result, the use of bipolar construction for microprocessors has been limited to sophisticated sixteen bit "number-crunching" machines, and to bit-slice elements, in which the disadvantages - particularly the high cost - are offset by the need for very fast operation.

More recently, Texas Instruments have introduced a new bipolar technology - integrated injection logic (I^2L) - which is claimed to combine most of the strengths of current MOS and TTL circuitry. TI are centring their efforts in the microprocessor market around the use of this new method of manufacture, but it is as yet too early to tell how significant this latest development will prove to be.

The possibilities represented by the microprocessor concept were very quickly apparent to the semiconductor manufacturers, and few if any of the major firms have failed to either develop and market their own device, or else make arrangements to second-source an already successful product. As a result, an extensive variety of devices are currently available.

Table 1 is an attempt to summarise the primary-source machines on the market. Progress in this field is too rapid for any such summary to be entirely accurate or up-to-date, but most commonly encountered devices are included.

An important aspect of the microprocessor market is Second-Sourcing. Almost all the most popular devices are manufactured, under agreement, by several firms other than the originator. From a practical point of view this may be useful to the purchaser, in offering greater security of continuous supply. However, it is also a psychological encouragement, endorsing the manufacturer's determination to support the product well into the future.

2.3 Eight-bit microprocessors

Eight bit microprocessors are by far the most widely used type, because of the combination of considerable flexibility with relatively low cost which they offer. A variety of devices is available, which exhibit a diversity of system structures; this variation may make a particular machine more suitable for a given task. As a result, it is worthwhile examining several representative systems individually.

(a) Intel 8080A. Intel are the pioneers in the microprocessor field; they were responsible for the first single IC MPU, the 4004, which was originally designed as a versatile calculator chip. They also produced the first 8-bit device, the 8008, and later supplanted it with the more advanced 8080.

The 8080 is frequently described as the "industry standard" which is an exaggeration; mass production processes are not well suited to the rapid introduction of revolutionary techniques, which means that microprocessors have yet to be adopted commercially on a large scale. The 8080 is popular because of its headstart; its comparatively long history favours a reputation of reliability, and Intel's experience in the field allows them to provide extensive hardware and software support to their customers. Compared to some of its competitors, the 8080 is technically unsophisticated, and has inherent drawbacks which later designs avoid. However, in practical terms it is as attractive a proposition as most of its rivals.

The system structure of the 8080 varies from the general description given earlier, in that the functions of the MPU are shared between two integrated circuits. In addition to the 8080A MPU chip itself, there is an 8228 system controller/bus driver device, which provides suitable external control signals and drive capacity to interface the MPU with its peripheral devices. In most microprocessors, these functions are built into the MPU itself, which may be seen as design progress since the inception of the 8080. It is doubtful whether this slight anachronism is of any real significance.

The 8080A system structure is shown in Fig 2. Like most other 8 and 16 bit microprocessors, the 8080 has an address bus 16 bits wide, allowing it to access the equivalent of 2^{16} = 64K of memory locations. The 8080 has an instruction set of 78 codes, among which are instructions allowing decimal calculations, and processing of 16-bit data. Depending on their complexity, the instructions take between four and eighteen cycles of the clock to be completed; each clock cycle is approximately two microseconds.

Processing is performed in an 8-bit accumulator, and six supplementary 8-bit registers which may be referenced individually or as pairs. This particular architecture is one of the advantages of the 8080, in that it provides considerable flexibility from simple programs; many straightforward tasks can be accomplished without using RAM for data storage.

All memory access is via absolute addressing modes; there is no facility to supply relative addresses, for example, in the form of a displacement from the current location. This can be a definite disadvantage, since performing modifications to a program, or relocating it in a different area of storage, require all specified addresses to be altered appropriately.

(b) Motorola MC6800. Motorola, although a relatively young company, have a very large share of the current semiconductor market, which means they have a strong reputation for supplying to the mass production customer. It has recently been announced that Motorola have contracted to supply microprocessors to the General Motors corporation for inclusion in their future automobiles - probably the largest single market in the world today.

Development of the Motorola microprocessor, the MC6800, was completed considerably later than the 8080, and as a result it has yet to establish an equivalent share of the market. However, the leeway is rapidly being reduced, since the MC6800 is accepted to be a more elegant and sophisticated machine.

The structure of the MC6800 setup is straightforward, and conforms to the general architecture shown in Fig 1. The MC6800 MPU device is fed with timing signals from an MC6870 crystal-controlled clock; the three buses interconnect the MPU to its peripherals. ROM is supplied by MC6830 chips of 1K byte capacity, while the MC6810 provides 128 bytes of RAM. Parallel I/O requirements are satisfied by the MC6820 peripheral interface adapter (PIA), and communications facilities are available via the MC6850 USART. A further device, the MC6860 modem, is designed to interface the system to standard serial communications channels such as telephone lines.

The MC6800 shares most of its operating features with the other common 8-bit microprocessors. It features only two working registers, compared to the 8080's seven, but it has more useful memory addressing modes which may compensate. All MC6800 peripherals, including I/O devices, are referenced as addresses in memory; this is a simple system to work with, although it may sometimes be uneconomical in terms of program space.

The MC6800 MPU requires only a single +5 V power supply, whereas others such as the 8080A need +5 V, +12 V and -12 V supplies. However, it must be pointed out that this is only advantageous if the remainder of the microprocessor system has the same facility. It is common for MC6800 systems to include ROM devices which require these three supplies, in which case the advantage is lost.

(c) Fairchild F8. The approach adopted by Fairchild in the design of the F8 exhibits a different philosophy to the two previous machines. Whereas the 8080 and MC6800 systems distribute the particular tasks - processing, memory, I/O etc - each to an individual device, the F8 system is an attempt to produce a simple microprocessor for dedicated applications in a minimum number of devices. A working F8 microprocessor can be configured in only two integrated circuit chips, together with a few passive components.

Inevitably, the F8 minimum system has severely limited capabilities, and the unusual architecture makes extension of the system complicated. Thus, an F8 microcomputer has no real advantages over an 8080 or 6800 based system of similar capacity, and may in some ways be inferior. The philosophy built into the F8 is ideally suited only to very straightforward microcontrollers, but this is certainly a very large potential market for microprocessor use. This fact is recognised in more recent developments by other big manufacturers, many of whom are now producing a simple, minimum system microprocessor suited to this role, in addition to their more flexible machines. The Intel 8048, and the Motorola MC6600 are examples of this type of system.

The F8 minimum system, as shown in Fig 3, consists of a 3850 central processing unit, and a 3851 program storage unit. The 3850 differs from the conventional MPU in several important respects. Most obvious are that it has built-in memory and I/O facilities; contained within the device are 64 bytes of "scratchpad" RAM, and two programmable parallel I/O ports, each eight bits wide. However, it does not include any of the registers or associated circuitry for referencing external memory; the program counter, stack register, etc, are constructed in the 3851 device. The PSU primarily consists of 1K bytes of ROM, which are used to store the program, but in addition it also provides the memory processing facilities, and two further eight-bit I/O ports.

Thus the minimum system provides processing unit, 1K of program store, a small RAM for storage of intermediate results, subroutine addresses, etc, and 32 I/O lines. For many dedicated tasks, such a setup is ideal. Some extensions - greater program space, more I/O capability - may be added by simply including further 3851 chips in the system. The devices have been designed to require a minimum of external circuitry to configure a working system; for example, clock circuitry is included within the 3850, and requires only an external crystal - or, if timing is not a critical factor, a simple RC circuit may be used instead. All the F8 system units work from +5 V and +12 V power supplies.

The F8 architecture does have limitations, the most obvious being the small area of RAM storage available; for any application involving large amounts of stored data or results, additional memory must be added. Since the 3850 does not contain its own memory control and addressing circuitry, an extra unit must be employed to interface the CPU to conventional RAM devices. This function is provided by a 3852 dynamic memory interface device, or a 3853 static memory interface, depending on the type of memory used.

The way in which functions are distributed throughout the F8 system has the drawback that it is more difficult to include in a particular configuration only those elements which are actually needed. A CPU/PSU combination has 32 I/O lines automatically, whether they are required by the system or not. While this is not in itself harmful, it has resulted in each device losing some individual power in its intended role; the 3850 is accepted to be an MPU of rather less processing power than its rivals. This limits the potential of the F8 microprocessor to the more straightforward tasks.

(d) Zilog Z80. The significance of the Z80 is that it is the newest, and hence the most sophisticated 8-bit microprocessor on the market. Developed by the designers originally responsible for the Intel 8080, it is claimed to be as great an advance compared to that machine, as the 8080 was compared to the 8008.

The architecture of the Z80 is conventional, resembling the MC6800. The internal structure is an improved version of the 8080A, featuring two sets of seven working resisters, which may conveniently be exchanged during program execution. The Z80 instruction set consists of 158 distinct instruction types, among which are all 78 of the 8080 instructions. As a result, programs written for the Intel device will run on the Zilog, although shorter, more powerful programs may be composed using the Z80 codes. This feature, combined with a much faster instruction time - typically 2 μs compared to the 8080A's 10-20 μs - results in a considerably more versatile and economical system. For users whose experience lies with the Intel machines, the most logical next step is likely to be adoption of the Z80 system.

2.4 Hardware development

When the semiconductor manufacturers first introduced microprocessors to the market, it was understandably on a device-orientated basis; they were accustomed to selling integrated circuits which purchasers built into systems of their own design, and microprocessor elements were not regarded in any different light.

However, microprocessors are systems, and their components are designed to be assembled in particular configurations. The design and construction of a working microprocessor, although straightforward in theory, may well be a time-consuming and problem-fraught process to the first time user who is unfamiliar with the concepts involved. Furthermore, expert designers of hard-wired logic are frequently unacquainted with software methods, which are so vital to the microprocessor.

As a result, the manufacturers - while still supplying microprocessor ICs on a large scale where required - have found it profitable to supply ready-built microprocessor systems, specifically for evaluation and development work. To the user, seeking a painless way of entering the field, this is by far the most satisfactory method; his first machine comes "consumer-packaged" with the initial work completed by specialists.

The systems available fall into two distinct categories:-

(a) Development systems are provided for the customer who has decided that he is ready to utilise microprocessors on a large scale, either in terms of mass production numbers of small machines, or as a few highly complex and powerful systems.

(b) Evaluation systems are intended to cater for the customer who has little knowledge of microprocessors, and is not prepared to make a large investment until he is convinced that microprocessors will be useful to him.

Inherent in the concept of either system is the inclusion of a Monitor Program. This will take the form of a standardised ROM, which allows the user to communicate with the machine. Under the control of the monitor, the system is capable of responding to simple commands input from a medium such as a data terminal, to provide basic functions. From the terminal the operator can place a program of his own derivation into RAM within the system, and run the microprocessor under the control of this program. The monitor provides simple facilities for fault tracing and rectification within the software. In this way, the operator can ensure that his program works as required, before he arranges for it to be set into ROM.

The user's programs must be re-entered into the system each time it is powered-up, but the facilities of the monitor are permanently available. It is this built-in software capability - sometimes termed "firmware" - which makes the microprocessor such an attractive proposition.

(a) Development systems. Most of the manufacturers of eight-bit microprocessors offer some type of development system - most of which are very similar. The most popular types are listed in Table 2.

In each case, the basic unit is very similar; a chassis consisting of a card-cage or a simple backplane, with a built-in power supply which will provide all necessary low-voltage DC supplies from a single mains connection. Also installed will be a few essential control switches - on/off, reset, etc - and suitable plug/socket connectors for direct interfacing with a Teletype or other communications terminal.

The card-cage unit allows the user to build up whatever system he requires from several standard or customised printed circuit boards. Normally the basic unit will include one or two cards, providing a minimum system; this will generally consist of an MPU with timing and buffering facilities, the minimum amount of RAM, and pre-programmed ROM containing the monitor program. For a development system, the monitor will normally be quite large and provide sophisticated functions. I/O is provided by a USART device.

Such a minimum system is ready for use, requiring only connection to a mains supply and a communications terminal. However, it is only really useful after the installation of several other cards. Large memory capacity may be added in the form of cards providing 8K or 16K bytes of RAM, suitable for holding large programs during development. I/O facilities may be extended with cards carrying several buffered PPI chips; generally a card will provide 40-48 programmable lines.

A development system provides a modelling environment in which a working setup may be evolved by the user, which closely resembles in both hardware and software, the eventual completed product. The versatile debugging capabilities enable a manufacturer to produce a thoroughly tested prototype well before committing himself to the expensive process of tooling-up for mass production. Once into production, the development system is by no means obsolete; it may be used, with whatever addition or modification may prove necessary, for the development of future products.

Other applications of a development system are conceivable, in which the unit might well be useful in its own right, rather than as a means to an end. For example, in a laboratory environment a great many processing, control, data handling, and communications tasks could conveniently be performed by such a machine, whose hardware and software configuration could be varied as required.

Depending on the capacity of the assembled system, a development system will cost the purchaser £2000-£5000; a communications terminal is essential, and this will add another £1000-£3000 to the bill if no suitable machine is already available. Thus, this is not a cheap solution; but it is likely to be a profitable investment for any purchaser anticipating an extensive and serious use of microprocessors.

(b) Evaluation systems. The appearance of microprocessors in a blaze of manufacturers' publicity evoked a predictable response from the world of science and technology - curiosity and cautious interest. Inevitably, faced with a new field of such widespread potential, and with a corresponding leeway for mistakes, few interested parties were ready to invest large sums in the acquisition of a full development system, or else to commit themselves to the time and effort required to produce their own working system from a collection of integrated circuits. The manufacturers' answer was to market evaluation systems.

Designed as they were to fill a well-defined need, the evaluation systems offered for the various microprocessors varied in little more than details. They universally consisted of a microprocessor minimum system configured on a single printed circuit board - MPU with clock and associated circuitry, PPI, USART with baud-rate generator, a small amount of RAM, and a monitor ROM.

The monitor program provided with an evaluation system is relatively limited in its capabilities, enabling the user to communicate with the system via a Teletype or similar terminal, and to write and run simple programs in the system RAM. The functions normally available include:-

- a command to insert numerical codes, instructions or data, into selected memory locations,

- commands to examine the contents, and where desired to change the contents, of the working registers or specified locations in memory,

- a command to print out the contents of a series of selected RAM location; on an ASR, this command may be used to produce a punched tape of a completed program for permanent storage. It should be noted, however, that the format of the tape will not necessarily be suitable for reloading into the microprocessor using the "insert" command.

- a command to commence operation at a specified memory location, which may be used to run users' programs.

Evaluation systems may be purchased either as an assembled and tested board, or as a kit to be put together by the user (some systems are available only in one form or the other). Inevitably the kits are cheaper, although this saving must be weighed against the need to assemble the machine, a process which is generally straightforward but time-consuming, and introduces the risk of possible malfunction due to constructional error. Low voltage power supplies are not included with the system, and must be separately provided by the user. This is rarely a problem, as suitable supply units are inexpensive and widely used. Communication, however, often proves to be a difficulty. Manufacturers have normally designed their evaluation systems - both hardware and software - to interface with a standard ASCII-coded serial terminal. Such a terminal provides excellent facilities - at considerable expense. If a machine is not already available, the acquisition of a new terminal is likely to cost in excess of £1000. This may well be unacceptable to the purchaser of an evaluation system costing £100-£350.

Realisation of this problem has caused several manufacturers to recently produce alternative versions of their evaluation systems, with simple built-in communication facilities. These normally consist of the standard board, with a ROM providing a different monitor program; a 4 x 4 keyboard allowing input access in hexadecimal form; and a readout display comprising several LED units. A further refinement is an interface allowing bulk storage and reloading of memory locations - particularly users' programs - on an inexpensive cassette tape recorder. Such systems represent an easy and cheap way of learning the rudiments of microprocessor usage; for any serious work, their capabilities are even more limited than the alternative evaluation system plus terminal combination.

Also now available are an enormous assortment of evaluation and/or development systems manufactured independently by smaller firms, from components bought from the semiconductor manufacturers themselves. These "middle-man" products are so numerous and varied that they defy generalisation; the spectrum runs from ultra-simple systems similar to, or even undercutting, the manufacturers' evaluation kits, to relatively sophisticated microcomputers. The hardware and software configuration of these systems may be as developed by the semiconductor manufacturer, or by the system vendor, or a combination of both. In many cases, these products represent excellent value for money; however, they are unlikely to have the in-depth support which the large manufacturers can offer.

One other distinct type of evaluation system which is available is the unit designed and built by a firm around the microprocessor components which they second-source. The philosophy involved in such a machine may differ widely from the primary manufacturer's similar system, which may in some cases be advantageous. For example, Mostek second-source both the Fairchild F8, and the Zilog Z80. The Mostek evaluation system for the F8, the "Survival Kit", differs in several important details from Fairchild's Evaluation Kit, while their SDB-80 board, incorporating the Z80, is totally different from Zilog's own Z80-MCB board.

Appendix is an attempt to summarise the significant features of several well-established evaluation systems, marketed either by the primary, or major second-source manufacturer. Due to the rate of progress in this field, such information cannot be entirely accurate or up-to-date; prices in particular are constantly changing, and are quoted only for comparison purposes.

Several considerations are important when choosing an evaluation system. The various MPU devices have some significant differences, which may make a particular device more attractive for some applications. However, unless the requirements of the situation are very closely defined, it is unlikely that the average purchaser will find detail discrepancies meaningful. Most experience shows that the time consumed in hairsplitting comparisons is better spent in actual work on any reasonably suitable machine.

When buying an evaluation system, the likely future course of the project is worth considering. Evaluation systems may be divided into two classes; those that may easily be extended to any required degree, and those that are more limited. If the tasks to which the microprocessor is to be applied are only defined in the very broadest terms, it is clearly more sensible to opt for an expandable system, which may subsequently be modified to suit any requirements. Conversely, if a more exact idea of the likely needs of the problem is available, the most efficient - and cheapest - machine will have just sufficient facilities in its own right, with no expansion necessary.

The decision to buy a machine either in kit form, or assembled and tested, must be made in the basis of time and facilities available. Assembly of a kit will present few problems to any electronics workshop, and may assist understanding of the principles embodied in the system; however, a guaranteed working unit may be a more attractive proposition in many circumstances.

Most of the remainder of this report will be concerned with the operation of the Intel SDK-80 machine which was obtained for study. The reasons which led to the choice of this system, rather than one of the many alternatives, may serve as a helpful guideline to some of the considerations involved.

Detail differences between the MPUs themselves were not significant, since none of the applications anticipated necessitated critical performance characteristics. Some uncertainty in the exact nature of the applications demanded a degree of flexibility, which led to the simplest and cheapest of the machines - the SC/MP - being discounted due to its limited potential. No effective distinctions could be drawn between the other devices on this basis; the Z80 was not available at that time, or its greater capabilities would have been an obvious point in its favour.

The relatively simple nature of the tasks to be performed by the machine meant that large expansion capacity was unnecessary; however, a very small system was likely to be inadequate, for which reason the MEK6800D1 kit was eliminated. (The MEK6800D2 was not on the market at the time; the absence of the need for a terminal would have been a strong advantage for this system.) Similarly, the unusual architecture of the F8, together with the limitations of the evaluation boards on sale, cast doubts on its long-term utility.

The RCA CDP18S020 was technically advanced, and featured the advantages of C-MOS circuitry. The single power supply, and the capability for on- and off-card expansion, were encouraging details. However, the specialised C-MOS RAM devices required were costly, and would have made an effective system prohibitively expensive. It was also noted that the RCA microprocessor was the only device under consideration which was not second-sourced.

The SDK-80 seemed to offer marginally the best combination of features available at the time. Although fairly expensive, and requiring three power supplies, it offered a useful system, easy expansion on a limited scale, and thorough hardware and software support.

2.5 Software development

Like any electronic device, a computer accepts a specified set of inputs, and produces a definite output in response to each. The computer has two distinctive features:-

(a) The large number of different possible input/output combinations,

(b) The ability to interpret sequentially a series of previously stored inputs (the program) at a speed limited only by its rate of operation.

Inputs must be presented on a group of parallel lines, each of which may be in one of two states (signal/no signal). Since most computer work is concerned with arithmetic manipulations, it is therefore conventional to regard the machine as handling binary numbers of a particular "width" - eg 8-bits, 16-bits, etc. A stored program consists of arrays of these binary numbers, each of which is accessed by the processing unit in turn, and understood as a particular instruction.

Programming is the process of producing appropriate stored programs - the Software - to obtain the desired operations from the computer. The end result of programming is a set of binary numbers which are understood by the machine; the programmer's task is to arrive at this point, having begun at a description of the problem to be solved, in terms of a human language and mathematics.

The translation process involved may be performed in several different ways; the trend is for increasing amounts of the work to be done by the machine itself.

Early computers were programmed directly in binary, with all translation being carried out manually; this resulting software is incomprehensible to a human reader, consisting as it does only of strings of ones and zeros. Deciphering a program is almost impossible, and mistakes very easy to make. Some improvement can be made by representing each binary number by its equivalent in another, more acceptable, base; the transformation can be simply performed by the computer. Decimal numbers, although preferable for the human operator, are difficult to interchange with binary, and it has become conventional to utilise Octal numbers (digits 0 - 7) or Hexadecimal numbers (digits 0-9 plus letters A-F). This gives a string of numbers more palatable to the user, but still obscure in their meaning. To transform a solution in conventional English/mathematics directly into these numbers is a painstaking and complex operation.

The popular method is to employ an intermediate language, a convenient mid-way point between human languages and machine code. There are two categories of intermediate languages:-

(i) Low Level (or Assembly) Languages,
(ii) High Level Languages.

Low level languages bear a close correlation to machine code; they consist of a number of code words - usually mnemonics - which have a one-to-one correspondence with the instruction set codes. For example, if the command "and the contents of a register B to the contents of register "A" is required, it is more convenient to write down "ADD B" and subsequently translate this into the corresponding instruction in machine code, than to immediately and obscurely put down "80(hex)". The programmer using an assembly language works close to the system structure, and is conscious of every detail of how the task is performed. It is necessary to specify memory locations exactly, although this is normally simplified by the use of relevant Labels; for example, an instruction such as "JMP PROG1" might be used to cause an unconditional branch to a program routine labelled "PROG1", which would be replaced during assembly into machine code by the appropriate start address.

Having written the program in low-level language, it must be assembled into machine code. This may either be performed manually, a straightforward if tedious process, or by the computer itself using a program termed an Assembler. This will accept the program as mnemonics and descriptive labels, together with a few instructions such as the start memory address, and will produce the corresponding machine code.

High level languages have a far closer resemblance to human language, particularly mathematical expression. For example, two numbers may be added together in the popular FORTRAN programming language by a statement such as "X = A + B". Most of the painstaking details which must be included in the eventual machine code, such as storage allocation, number magnitude handling, and data transfer, are dealt with during the translation process. This is performed by a complex program termed a Compiler. High level languages are advantageous for writing long and involved programs, and are particularly suited to application by inexpert users; languages are machine-independent, and it is the task of the compiler to generate suitable machine code for that particular computer. Because of the flexibility which must be built into a high-level language, the resulting machine code is unlikely to be as concise and efficient as that produced by assembler language coding, which tailors the program exactly to the system requirements.

The programming methods adopted for use with microprocessors are derived from traditional computer practice, but within the restraints applied by the peculiarities of microprocessor systems. The simplest minimum systems normally provide the capability to load machine code in an octal or, more commonly, a hexadecimal form. This requires that the program should be written in this form - usually impractical - or else written in low-level language, and then assembled manually. For short programs, this is not a particularly inconvenient process, although mistakes can be made and may be awkward to eradicate.

The use of an assembler program makes programming a great deal easier, but places certain requirements upon the system. A commercially available resident assembler to be operated on an 8-bit microprocessor will commonly need 8K bytes of memory available to hold the assembler itself, together with the assembly language and machine code versions of the user's program.

One alternative is to use a Cross-assembler, a program which runs on a separate mainframe or minicomputer, and may be used to prepare software for the microprocessor. The viability of this approach depends, of course, on the availability of computer time; in the absence of an in-house machine, it is possible to arrange to use most popular cross-assemblers on commercial time-sharing systems via a telephone link. This can prove to be a costly method if a great deal of traffic is anticipated.

The use of high-level languages in combination with microprocessors must be considered somewhat unusual, and of doubtful value. The powerful programming applications which require the use of high level techniques are probably ill-suited to running on a microprocessor, and would be better located on a machine of greater capacity. The inefficient machine code output from a compiler is undesirable on a small machine with very limited memory storage; moreover, cross-compiling is almost inevitable, since a resident FORTRAN or BASIC compiler for a microprocessor will occupy far more RAM than can otherwise be justified.

Microprocessors are very dependent on the concept of Firmware - the storage of completed software in a non-volatile form in ROM. For any program which is likely to receive extensive usage, it is worthwhile ensuring continuous, immediate access, by the use of ROM storage. Machines are available for in-house PROM programming; and most semiconductor dealers now operate return-of-post PROM programming services, for £2-£3 per chip.

3 INTEL SDK-80 MICROPROCESSOR SYSTEM HARDWARE

3.1 Power supply unit

The Intel SDK-80 microprocessor board is designed to operate from three low-voltage DC power supplies, as detailed below:-

Supply voltage	Current requirement	
	Minimum system	Maximum system
+5 V	1.3 A	2.1 A
+12 V	0.35 A	0.45 A
-12 V	0.2 A	0.3 A

A suitable power supply unit was assembled, incorporating three ITT Powercards, each with a rated 15 W output; two types were used:-

Power unit type	Supply voltage	Maximum current
PC 1000 A15	12-15 V	1 A
PC 3000 A5	0-5 V	3 A

Using the built-in adjustment potentiometers, the output supplies were adjusted to +5 V, +12 V and -12 V with respect to a common earth. Supply outputs were taken from three, 7-way MRE (pattern 103) sockets, installed on the front panel and connected in parallel. This enabled the unit to be used as a general laboratory electronics power supply, as well as its use for supplying the microprocessor and its associated interface circuitry. The pin designations of the 7-way output sockets are shown in Fig.4a.

3.2 Communications terminal

Like most microprocessor evaluation systems, the SDK-80 is designed to communicate via a data terminal. Several types are available, ranging from electromechanical devices such as the Data Dynamics ASR 33, to the more modern, sophisticated machines like the Texas Instruments Silent 700, and prices similarly vary in the range £1000-£3000 for a new machine.

The ASR terminal provides the following facilities:-

(i) Manual input from the keyboard,

(ii) Automatic input from the punched-tape reader,

(iii) Output from the printer and, where required, from the paper tape punch.

The maximum rate of data transfer is 10 characters per second (110 baud). Data is transferred in serial form, using the 8-bit ASCII code.

Some consideration was given to achieving the required input/output functions without a terminal, using separate devices such as a paper tape reader, tape punch, hexadecimal keyboard, LED displays, etc. Such an arrangement would be far less convenient to commission and use; it offers little advantage in cost over a terminal, and indeed is inferior in this respect if the machine can be purchased secondhand for approximately £400.

The only shortcoming of the machine actually purchased, a Data Dynamics ASR 33, was that it did not feature the automatic start/stop facility on the paper tape reader, a function found on some later models; this would have been ideal for microprocessor control, resulting in a considerably more versatile system.

3.3 The Intel SDK-80 evaluation kit

The SDK-80 microprocessor board may only be purchased in kit form, to be assembled by the buyer ; had an assembled and tested version been available, this would probably have been preferred. However, assembly proved to be a very straightforward process, and the instructions supplied were sufficiently simple and exact that little or no knowledge and experience of electronics work was required.

The board may be configured to accept input communications signals in either of two distinct formats, as a voltage level, or as a 20 mA current loop, by means of variable "jumper" connections. The terminal was of the voltage level type, and the appropriate jumper connections were made on the SDK-80.

The SDK-80 board is ready configured to take a more extensive system than that comprised by the components supplied in the kit. In particular, the amount of RAM may be increased from 256 bytes to 1K bytes; and an extra 8255 PPI device may be installed, increasing the number of I/O lines from 24 to 48. The integrated circuits required for these extensions were ordered with the kit itself, and were installed straight away. For convenience, all chips were installed in IC sockets, rather than by direct soldering.

3.4 Assembled microprocessor system

The completed SDK-80 board was securely fitted into a Lektrokit box of suitable size, with power supply lines brought in through one side. All connections with the board, with the exception of the power lines, were routed via the three 25-way Cannon plug/socket assemblies on the edge of the board. This enabled the front panel to be completely separated from the microprocessor unit, when required.

Facilities for possible input/output connections were provided with four 18-way MRE (pattern 103) sockets, installed on the front panel. Also fitted were eight BNC sockets, intended to allow individual control lines to be taken from Port C of the user-supplied 8255 chip (USC). No connections were made committing these I/O sockets until they were required. Subsequent work made it necessary for the microprocessor to be connected to an external shift register unit: eight data lines, two control lines, and an interrupt control line were routed via one of the 18-way MRE sockets, according to the pin designation format shown in Fig 4b.

Also installed on the front panel are a 25-way Cannon socket for connection of the terminal; the RESET pushbutton; and the interrupt control facilities, consisting of an INT REQ pushbutton, and two BNC sockets providing an $\overline{\text{EXT INT}}$ input and an INT ACKNOWLEDGE output, respectively. Additional interface circuitry, as shown in Fig 5, is installed in the wire-wrap area of the SDK-80 board, most of which is required to provide suitable interrupt facilities. An input is available at pad H on the board, $\overline{\text{INT REQ}}$, which when grounded causes a hardware-generated CALL instruction to be inserted into the program. This gives rise to an unconditional jump to 13FDH, a location in RAM, at which point a further JUMP instruction may be inserted by the user. By the use of this system, the microprocessor may be allowed to continue with operation of a program, but alerted immediately if some other course of action - for example dealing with intermittent input of data - should be required. Using the additional circuitry, an interrupt sequence may be triggered by:-

(a) a logic "1" being presented on lines CØ or C3 of either PPI device, possibly as part of a "handshaking" routine during data input,

(b) Manual operation of the INTERRUPT pushbutton on the front panel, which inputs a signal via a simple flip-flop, to eliminate contact bounce problems,

(c) A direct interrupt input from the $\overline{\text{EXT INT}}$ socket on the front panel,

(d) An input from an external source, via pin N of the connected MRE socket, and through a monostable multivibrator to ensure a reliable pulse width. This facility was provided as part of the interface with the shift register unit, as described above; operation of the RESET control on the shift register unit was used to resequence program running in the SDK-80.

Also as part of this microprocessor/shift register unit interface, lines C1 and C2 of the kit-supplied 8255 chip (KSC) were used as source control (SC) and acceptor control (AC) in a handshaking routine. The signals generated by the two systems were "upside down" with respect to each other, and it was necessary to include inverters (configured with NAND gates) in the lines.

3.5 Future system extension

While microprocessor systems are inherently capable of variation to suit the requirements of a particular task, it may not always be simple to extend an existing unit. The SDK-80 board is designed with a great deal of flexibility in the I/O structure, and by adding additional components in the wire-wrap area, the user can configure the system to interface with almost any peripheral device. However, the board offers very limited capability for any expansion of the microprocessor system itself, particularly in terms of increased memory capacity. Although most control signals are available at a number of pads in the centre of the board, the data and address buses are not easily accessible. The SDK-80 is obviously not intended to be used as the central element of a larger system, and suppliers advise against attempts at large-scale expansion.

However, it would appear to be straightforward to increase the amount of RAM from the existing 1 K bytes to 2 K bytes. The RAM consists of P2111-A4 chips (128 x 4 bit capacity) which are selected in pairs by means of an 8205 binary/one-of eight decoder device; there is thus the capacity to address eight pairs of RAM chips, of which only half is currently employed. Since the wire-wrap area is limited, it is likely to be more convenient to configure this extra 1K RAM on a separate board. The SDK-80 is designed for buffers (two 8212 devices) to be placed in the address bus; the data bus may be buffered simply with two 8206 (I/O port) chips, and any of the control lines may be extended through traditional TTL buffers.

The three buses may be extended onto the additional memory card, which would carry:-

(a) Eight P2111-A4 chips (1K bytes RAM),

(b) An extra 8205 binary/one-of-eight decoder, selecting memory locations 1400H - 17FFH,

(c) Any passive components (decoupling capacitors, etc) required.

Power supplies would be taken in parallel with the SDK-80 itself. The power supply unit described earlier should be just adequate to drive the maximum SDK-80 system, together with the additional 1K RAM card.

4 INTEL SDK-80 MICROPROCESSOR SYSTEM SOFTWARE

4.1 Operating the SDK-80 system

Operation of the SDK-80 system is straightforward, provided the procedures and instructions given in the "SDK-80 User's Manual" and "Intel 8080 Manual" are followed. From experience, two main points are worth mentioning:-

(a) At initial power-up, the terminal will print out the sign-on message, followed by the prompt character,

```
MCS-8Ø KIT
.
```

This is the normal response to RESET, and so it is easy to believe that the SDK-8 automatically comes up in the RESET condition at power on. This is not normally the case - the RESET button should always be operated after power-up, otherwise confusing problems are likely to be encountered.

(b) Having established that the system is in a RESET state, it is recommended that the first action should be the setting of the interrupt vector point, by typing:-

```
I13FD
C3 ∅8 ∅∅          (terminate with ESC)
```

This is necessary, since the interrupt system is connected. Once interrupts are enabled (following any "Gxxxx" command), operation is vectored to location 13FDH in response to:-

(i) An FFH code in the program, which is a common random value,

(ii) A ground pulse on the $\overline{\text{INT REQ}}$ line, which sometimes occurs due to mains noise.

The random values contained in locations 13FDH - 13FFH at power-up may give rise to surprising activities by the microprocessor, which in some cases - such as the disruption of user programs - may be very troublesome. A C3H (JMP) instruction to any convenient address (∅∅∅8H returns control to the monitor) will prevent any unexpected interrupt having harmful effects.

4.2 Writing programs for the SDK-80

The first stage in writing programs for the microprocessor is to study the problem to a high degree of detail, and to decide exactly how each process is to be carried out. Unlike the high level language and complex operating system found on most mainframe computers, the minimal Monitor program of the SDK-80 requires the user to be precisely aware of every detail of a program. There is much to recommend a subroutine-centred approach to programming; each individual task which must be performed by the finished program may be written as a separate routine, which may then be run, tested and corrected as necessary. A simple framework program may then be written which "calls" the subroutines in the appropriate sequence. There is the obvious advantage that a process performed several times may be accomplished simply by several calls to the subroutine in question; fault tracing is also simplified, since it is quickly possible to locate the area where the problem must lie.

In all but the smallest programs, it may well be helpful to produce a flowchart of the system to clarify the exact sequence of operation.

The program is written in Assembly Language mnemonics, with addresses represented by symbolic labels. It is good practice to adopt the format understood by the assembler program, even in the first draft; the listing of the monitor program, given in the back of the "SDK-80 User's Manual" is a good guide. It is worthwhile making the effort to put in a few comments as aids to memory on subsequent readings.

Having written the program, the method of assembly must be decided. For short programs, or small subroutines which will later be combined to form a larger program, hand assembly may be preferred. Since, using this process, the terminal is not used until the stage of entering the finished machine code, there is considerably less typing to do, with a consequent reduction in the likelihood of errors being introduced by inexperienced typists. However, error correction in hand-assembled programs must be performed at the machine code instruction level, which is extremely troublesome for all but the most minor mistakes.

In order to assemble the program using the MAC-80 Cross Assembler, available on most commercial timesharing systems, the program listing must first be prepared - in the form of mnemonics, symbolic labels, and comments - on punched paper tape. For this purpose, the terminal is used in "Local" mode; some of the more tedious tasks involved, such as the production of line numbers in the printout, may be alleviated with the aid of the microprocessor itself. A suitable program will exercise limited control over the terminal, making it "intelligent" enough to perform such details automatically. Producing the punched-tape listing is still time-consuming, but once completed and entered into a computer storage file, it may be manipulated and modified to any necessary degree, with a minimum of effort.

Hand assembly of programs is tedious, but quite easy, and highly accurate if a systematic approach is followed:-

Having completed the listing in mnemonics, the next stage is to perform the encoding into hexadecimal values; it is inadvisable to allocate memory locations at the same time. If this prevents addresses being completed in three-byte instructions, the resulting space should be clearly marked with an asterisk.

After encoding, the storage locations may be assigned, counting through sequentially, and marking the address at the beginning of each instruction (one, two or three bytes). Two extra locations must be allowed for each gap which has been left and marked with an asterisk; when the entire program is located, the missing addresses may then be appropriately filled in.

Assembly using the "MAC-80" Cross Assembler is performed as described in the "8080 Assembly Language Programming Manual" and the relevant guide to using MAC-80 on the timesharing service.

Hand assembled programs are manually typed into the microprocessor RAM, using the monitor "I" command. MAC-80 produces a punched tape of the assembled program, expressed as hexadecimal values in a particular format. A short loading routine may be written to run on the SDK-80, which allows this tape to be run in immediately. Once loaded, it is worthwhile obtaining a listing of the stored program straight away. There is then a convenient hard copy of the program to work on.

At this point the program may be run; probably some debugging will be needed before it will work correctly. Errors fall into two categories - programming errors, and coding errors. Fault-finding is performed using the facilities of the monitor program, primarily by setting "break points". Placing the "RESET 1" instruction (CFH) at any point in the program will cause control to be returned immediately to the monitor, all register and memory contents being preserved for scrutiny. These values may then be examined using the "X" and "S" commands, and hence checked against expected results at that point. By working sequentially through the program, trouble spots may quickly be isolated.

More sophisticated software systems than the SDK-80 Monitor commonly provide a "relocatable break point" facility, whereby the insertion of a break point relocates all subsequent instructions, so that the program sequence is not disturbed. This is not possible with the SDK-80; either spaces must be left at points throughout the program, or the break point must be inserted replacing existing instructions, and removed before continuing. The first course of action requires no-operation instructions (ØØH) to be placed at intervals throughout the program during encoding. This is slightly tedious, and uneconomical on RAM storage, but it does allow faster debugging.

Frequently, errors may only be rectified by the insertion of several instructions, for which there is unlikely to be space - even if NOP codes have been included. The only solution, short of rewriting the entire sequence, is to "jump" clear into an uncommitted area of storage, and proceed with the required additions, before jumping back into the main body of the program. A long program, composed of several sections debugged in this way, is likely to be a maze of apparently illogical branches and confused small routines, so it is vital to keep records of the program up to date, with all changes and additions marked. Programs which were originally assembled using MAC-80 are straightforward, since adjustments can be embodied in the assembly language file, and reassembled to produce a corrected machine code sequence. The finished version of a hand-assembled program is likely to be more untidy; any attempt to "streamline" all corrections into logical sequence within the program will probably result in the creation of further errors.

4.3 <u>General programming comments</u>

It is worthwhile becoming thoroughly familiar with the Monitor program supplied as a ROM with the SDK-80, of which a listing is given in the back of the "User's Manual". The majority of the Monitor consists of several short subroutines, each of which performs a specific, useful function. Since these routines may be called from the user's own program, they may be conveniently utilised within most applications. In particular, the routines handling I/O between the microprocessor and the terminal are frequently needed, and it is obviously more efficient - and convenient - to use the appropriate monitor routines, than to write a suitable alternative.

The subroutines, and also the larger subsections of program, found in the Monitor, are useful guides to the methods adopted in system programming. Other examples of programs written by Intel software experts may be found in the "8080 Assembly Language Programming Manual" supplied with the SDK-80 kit. Many of these routines, as written or with minor modifications, may be used to perform the desired functions within user's programs; provided extensive modifications are not required, such routines are worth using, since the professional software specialists are more likely to write efficient programs. If considerable alteration is necessary to obtain exactly the required results, it may be advisable to write a new program from scratch instead; most of the advantages of using an existing program are lost, and mistakes are likely to creep in due to incomplete understanding.

A frequent request from potential microprocessor users, is an approximate guide to the parameters involved in performing particular tasks with such a machine; for example, the RAM storage required for operation, the number of program instructions, and the time taken for completion, are of obvious significance. It is difficult to produce even a rough guideline for any particular task without actually producing a program, since the complications involved in solving most problems may not be immediately apparent. However, some general impressions may be obtained by examining the parameters found in several useful applications routines, listed in Table 3. These figures are all generalised estimates, based on the operation of the 8080A, which features a cycle time of 2 μs.

A particularly critical area in microprocessor programming is the handling of input/output operations. Communications via the terminal will normally be thoroughly covered by Monitor subroutines, which may be "called" up by user's programs, and thus few problems remain for the programmer. Little guidance is provided by the supplier, however, regarding I/O via the PPI devices. It is likely to be worthwhile for the user of a microprocessor system to experiment extensively with the various I/O modes available, to arrive at a thorough understanding of the peculiarities of operation. In general, the most important consideration seems to be that, however sophisticated the capabilities of the hardware, most of the work will need to be performed in the program itself. It is vital to know exactly how far the electronics can be allowed to "look after themselves", otherwise significant details are likely to be omitted from the software control.

5 CONCLUSION

There seems little doubt that the microprocessor will be an extraordinarily successful product, and will become more and more prevalent in many areas of application during the next few years. The pace of development in the semi-conductor business is extremely rapid, and the evolution of microprocessors since their inception has been very great. However, it would appear that the field has now been thoroughly explored by the manufacturers, and progress in the immediate future is likely to consist of variations of detail, rather than any significant revolutionary improvements. It seems certain that early fears that an "industry standard" would emerge, causing all other products to be discontinued as uneconomic, will not be realised; no single microprocessor has become entirely prominent, and a number of major manufacturers now have too large an investment in the field to lightly consider abandoning competition.

Among eight-bit microprocessors, there are a number of viable machines on the market. There are detail differences, which may make some better suited to particular applications, but for many purposes there is little to be gained by painstaking comparisons; any of the available machines is likely to be adequately suitable. Choice is best made on the basis of availability, hardware and software support, and similarly practical features.

Experience of the SDK-80 suggests that a microprocessor could be a very useful tool in many laboratories. Its application to a data processing role was encouragingly successful; its primary strength lay in the ability to handle large amounts of data automatically. However, such a machine is not well suited to extensive arithmetical manipulations, with routines for multiplication, division or square roots requiring large amounts of storage, and a great deal of running time. Any really involved calculations are probably best performed externally, with the microprocessor handling the initial data, and outputting intermediate results. It was apparent that, although principally obtained with a few particular tasks in mind, many other applications for the machine quickly became obvious once it was available. Programming the microprocessor is not a particularly rapid process, especially where more complex results are required; however, this software development is far quicker and easier than the implementation of the same functions using hard-wired logic. Except where very high speed is required, a microprocessor could profitably be used to replace most of the complex electronic systems employed in a wide range of equipment, and should be considered as a possible solution to any appropriate new requirement.

Appendix

MICROPROCESSOR EVALUATION SYSTEMS

MOTOROLA MC6800

Technology - n-MOS

Instruction Set - 72

Motorola MEK6800D1 Evaluation Kit;-

RAM - 256 bytes static

ROM - 1K bytes

I/O - USART; PPI giving 20 lines

Power supplies - +5V, +12V, -12V

Firmware - 1K Monitor

Communication - via terminal

Extension - limited area on card

Approx. Price - £85 in kit form (£130 including extras for completion)
£150 assembled and tested by dealer

Motorola MEK6800D2 Evaluation System;-

RAM - 1K bytes static

ROM - 1K bytes

I/O - USART; PPI; (both committed)

Power supplies - +5V, +12V, -12V

Firmware - 1K Monitor

Communication - via supplied I/O board; input from 24-key hexadecimal keyboard, limited output via 6-digit hex LED display. Bulk storage (programs etc) via USART to domestic cassette tape recorder.

Extension - limited area on board

Approx. Price - £450 for two boards, assembled and tested

NATIONAL SEMICONDUCTOR SC/MP

Technology - n-MOS

Instruction Set - 46

Nat Semi Introkit;-

RAM - 256 bytes static

ROM - 512 bytes

I/O - serial or parallel communications interface

Power supplies - +5V, -12V

Firmware - ½K Monitor (different versions for serial or parallel I/O)

Communication - via terminal or hand-held terminal (serial); via Nat Semi hand-held hexadecimal keyboard/display unit (parallel)

Extension - limited area on card; comparatively easy extension on extra cards

Approx. Price - £65 in kit form
£150 for hand-held terminal, or hex keyboard/display

FAIRCHILD F8

Technology - n-MOS

Instruction Set - 75

Fairchild Evaluation Kit:-

RAM - 1K bytes static

ROM - 1K bytes

I/O - 32 parallel lines

Power supplies - +5V, +12V

Firmware - 1K Monitor

Communication - via terminal

Extension - difficult

Approx. Price - £125, factory assembled and tested

Mostek Survival Kit:-

RAM - 1K bytes static

ROM - 1K bytes

I/O - 32 parallel lines

Power supplies - +5V, +12V

Firmware - 1K Monitor (significantly different from Fairchild version)

Communication - via terminal

Extension - difficult

Approx. Price - £100 in kit form
£125, factory assembled and tested

RCA CDP1802

Technology - C-MOS

Instruction Set - 91

RCA CDP18S020 Evaluation Kit;-

RAM - 256 bytes + 32 bytes committed to Monitor

ROM - 512 bytes

I/O - 8-bit input port; 8-bit output port; serial terminal interface

Power supplies - +5V; battery operation capability

Firmware - ½K Monitor

Communication - via terminal; continuous monitoring via on-card LED displays

Extension - extensive on-card expansion area, including pre-connected sockets for additional 4K RAM: all signals accessible for further expansion

Approx. Price - £180 in kit form

ZILOG Z80

Technology - n-MOS

Instruction Set - 158

Zilog Z80-MCB Microcomputer Board;-

RAM - 4K bytes dynamic

ROM - 512 bytes - 4K bytes

I/O - USART: PPI giving 2x 8-bit ports + control

Power supplies - +5V

Firmware - Monitor (½K to 3K versions)

Communication - via terminal

Extension - limited on-card area; buffered for extension on separate cards

Approx.Price - £410 assembled and tested (including 1K Monitor)

Mostek SDB-80 Software Development Board;-

RAM - 4K or 16K bytes dynamic

ROM - 2K - 20K bytes

I/O - USART: 2x PPI giving 4, 8-bit ports + control

Power supplies - +5V, +12V, -12V

Firmware - 2K Monitor; 8K assembler/editor

Communication - via terminal

Extension - buffered for expansion on separate cards

Approx. Price - £775 assembled and tested (including 4K RAM and 10K ROM)

__INTEL 8080A__

Technology - n-MOS

Instruction Set - 78

__Intel SDK-80 System Development Kit:-__

RAM - 256 bytes static

ROM - 2K bytes

I/O - USART; PPI giving 2, 8-bit ports + control

Power supplies - +5V, +12V, -12V

Firmware - 1K Monitor

Communication - via terminal

Extension - prewired for additional PPI, and increase to 1K bytes RAM and 4K bytes ROM; limited on-card area for expansion; further expansion onto extra cards difficult

Approx. Price - £250 in kit form.

IT
165

Table 1

POPULAR MICROPROCESSORS

Manufacturer	4-Bit	8-Bit	12-Bit	16-Bit	Bit Slice
Advanced Micro Devices					AM2901
Fairchild		F8			9400
Ferranti				F100-L	
General Instrument		LP8000		CP1600	
Intel	4004	8008			3000
	4040	8080			
		8048			
Intersil			IM6100		
MOS Technology		MCS6500			
Motorola		MC6800			MC10800
		MC6600			
National Semiconductor		SC/MP		PACE	IMP-16
Plessey				Miproc	
RCA		1802			
Rockwell	PPS-4	PPS-8			
Signetics		2650			
Texas Instruments	TMS1000			9900	SBP0400A
Western Digital				MCP1600	
Zilog		Z80			

Table 2

MICROPROCESSOR DEVELOPMENT SYSTEMS

Microprocessor	Development System
Motorola MC6800	Exorciser
Intel 8080A	Intellec MDS
Fairchild F8	Formulator
RCA 1800	COSMAC CDP
Nat Semi SC/MP	LCDS

Table 3

OPERATING PARAMETERS OF EXAMPLE ROUTINES

Function Programmed	Program Storage RAM/PROM (bytes)	Data Storage RAM (bytes)	Time Taken (μs)
Subtraction (16 bits - 16 bits)	8	-	30
Multiplication (8 x 8)	21	-	600
Division (32/32)	150	16	6000
Square Root (16 bit)	250	32	68500
Decimal/Hexadecimal Conversion (3 Decimal digits/8 bits)	95	-	750

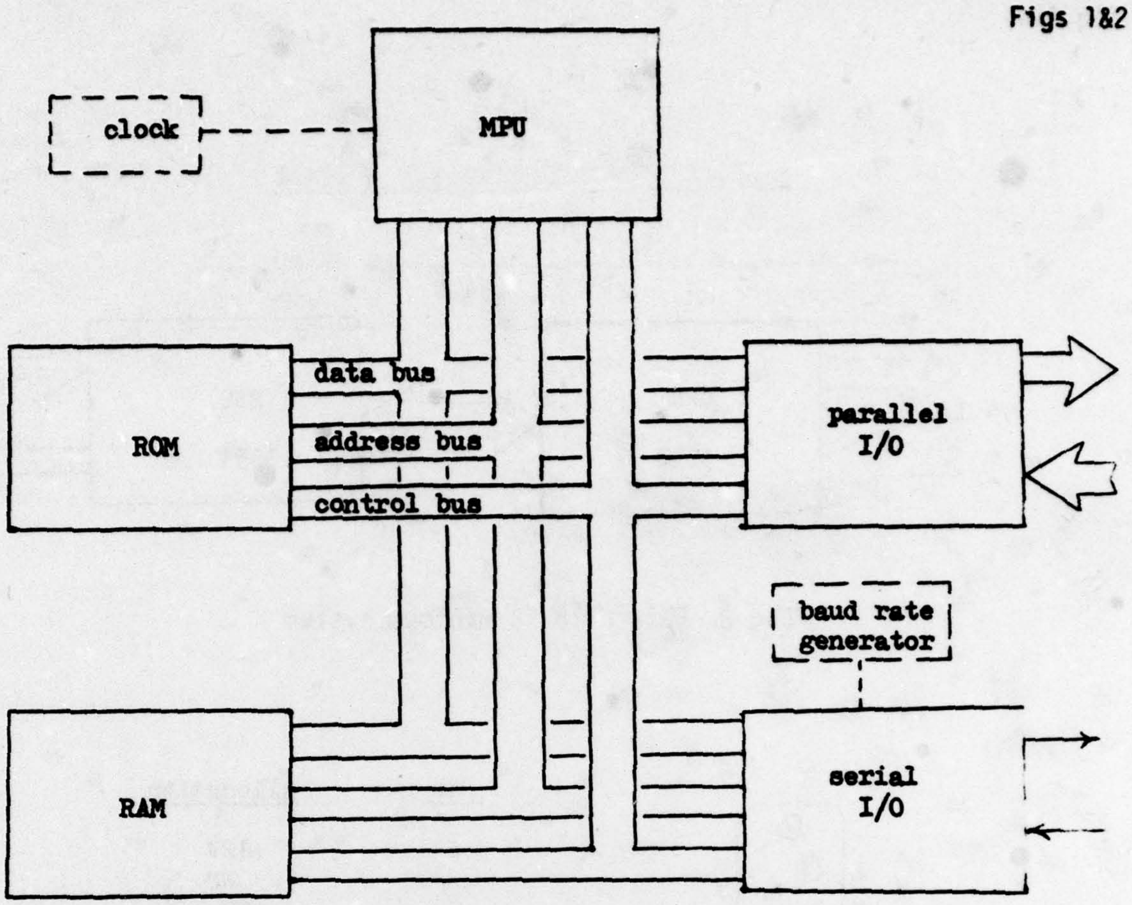


Fig 1 Generalised microprocessor system

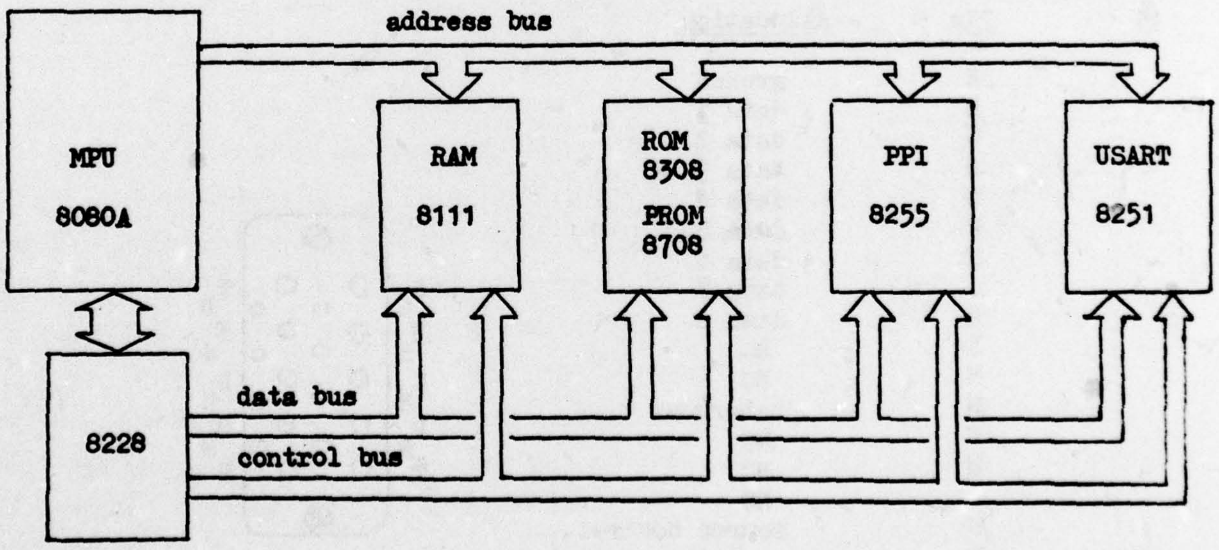


Fig 2 Intel 8080 microprocessor

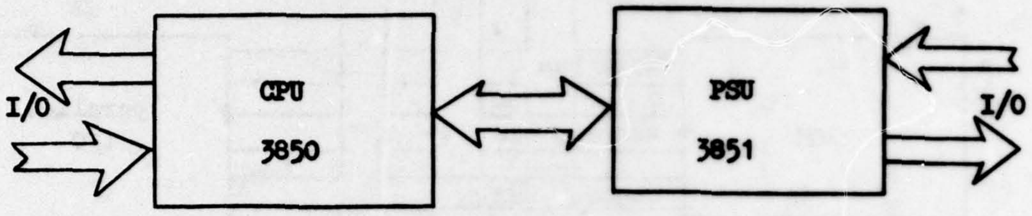
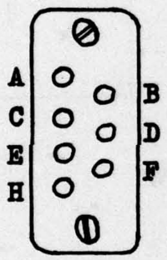


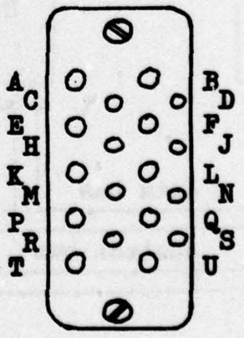
Fig 3 Fairchild F8 minimum system



Pin	Allocation
A	+12V
B	NC
C	+5V
D	NC
E	ground
F	NC
H	-12V

(a)

Pin	Allocation
A	ground
B	data 1
C	data 2
D	data 3
E	data 4
F	data 5
H	data 6
J	data 7
K	data 8
L	NC
M	NC
N	interrupt
P	NC
Q	NC
R	NC
S	source control
T	acceptor control
U	NC



(b)

Fig 4 MRE socket pin designations

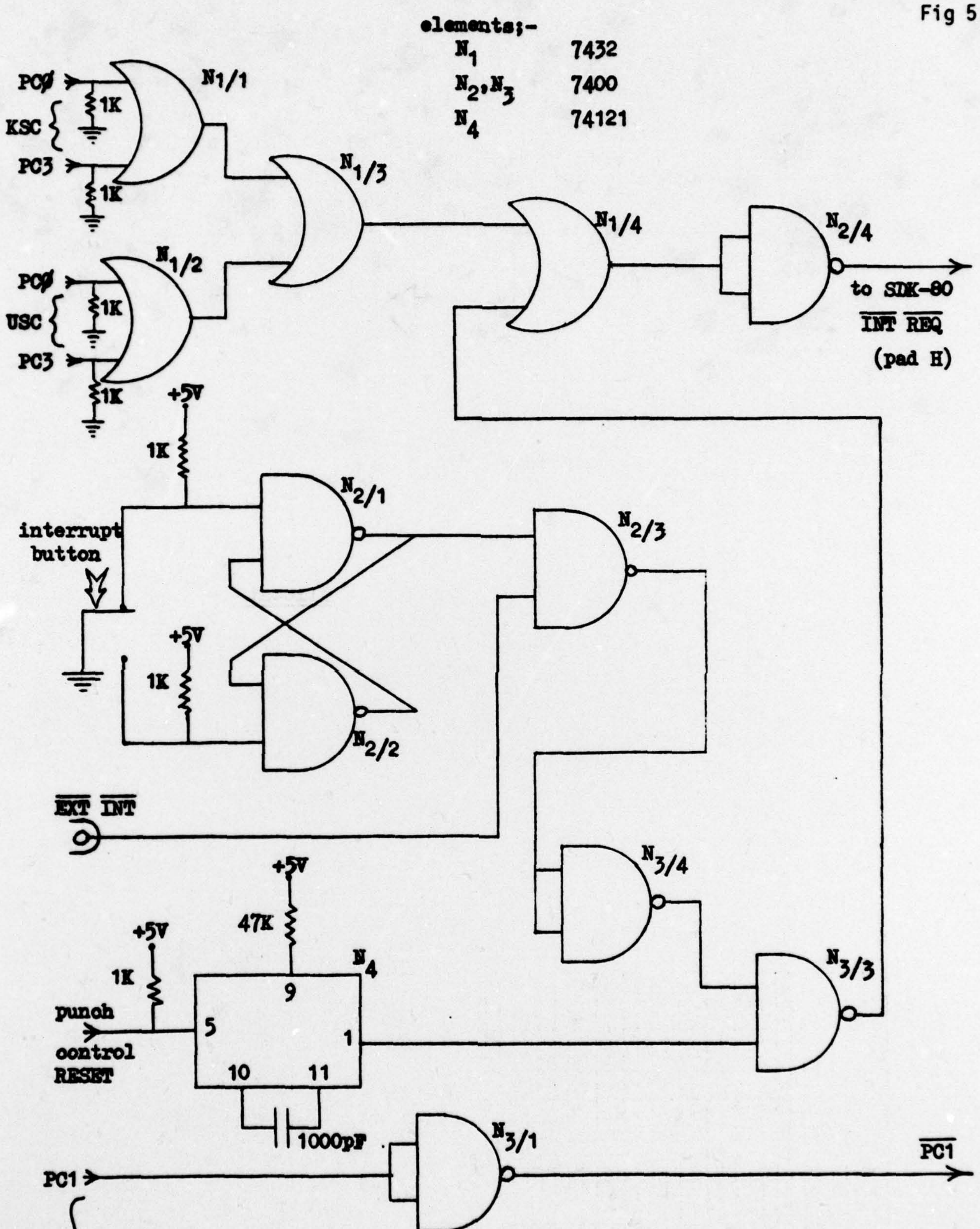


Fig 5 Additional interface circuitry on SDK-80 wirewrap area

REPORT DOCUMENTATION PAGE

Overall security classification of this page

UNCLASSIFIED

As far as possible this page should contain only unclassified information. If it is necessary to enter classified information, the box above must be marked to indicate the classification, e.g. Restricted, Confidential or Secret.

1. DRIC Reference (to be added by DRIC)	2. Originator's Reference RAE TM IT 165	3. Agency Reference N/A	4. Report Security Classification/Marking UNCLASSIFIED

5. DRIC Code for Originator 850100	6. Originator (Corporate Author) Name and Location Royal Aircraft Establishment, Farnborough, Hants, UK
5a. Sponsoring Agency's Code N/A	6a. Sponsoring Agency (Contract Authority) Name and Location N/A

7. Title
The microprocessor, and its application in a laboratory environment

7a. (For Translations) Title in Foreign Language

7b. (For Conference Papers) Title, Place and Date of Conference

8. Author 1. Surname, Initials Sockett, M.	9a. Author 2	9b. Authors 3, 4	10. Date August 1977	Pages 45	Refs. -
11. Contract Number N/A	12. Period N/A	13. Project	14. Other Reference Nos.		

15. Distribution statement
(a) Controlled by – UNLIMITED
(b) Special limitations (if any) –

16. Descriptors (Keywords) (Descriptors marked * are selected from TEST)

Microprocessors.

17. Abstract

This Memorandum records the experience gained in the selection and application of a microprocessor. Observations are included about microprocessors generally, with particular emphasis on eight-bit machines. The Intel 8080A device, and the Intel SDK-80 Evaluation Kit, are considered in detail.

1/01653

RAE Form A143